The evolutionary resilience of distributed cellular computing

The Evolutionary Resilience of Distributed Cellular Computing

Matteo Cavaliere[1] and Alvaro Sanchez[2]

[1] University of Edinburgh, UK,
mcavali2@staffmail.ed.ac.uk,
[2] Yale University, USA
alvaro.sanchez@yale.edu

Abstract. Individual cells process environmental information relevant to their functions using biochemical processes and signalling networks that implement a flow of information from the extracellular environment, across the cell membrane to the cytoplasm in which the actual cellular computation takes place (in the form of gene expression). In many cases, the environmental information to be processed are either molecules produced by other cells or shared extracellular molecules - in this case the processing of the environmental information is a distributed, highly parallel computing process, in which cells must synchronize, coordinate and cooperate. While the ability of cells to cooperate can increase their overall computational power, it also raises an evolutionary stability issue - population of cooperating cells are at risk of cheating cells invasions, cells that do not cooperate but exploit the benefits of the population. The bridge between membrane computing (as a mathematical formalization of cellular computing) and evolutionary dynamics (as mathematical formalization of Natural selection) could lead to interesting insights on the evolutionary stability of cellular computing.

1 How much cells can compute

Populations of living cells can be seen as systems that implement information processing in a distributed manner. This is a repeated motif in biology, where examples of biological systems that are distributed, process information and make decisions without a centralized coordinator occur at multiple levels of organization [10, 14, 16]. Several authors have tried to understand the similarities and differences between distributed computations in biological and human-designed systems (e.g., see [14] for a recent review). Distributed information processing in cellular populations is often result of the interplay between inter-cellular communication and cellular physiology. Cells can often secrete signaling molecules which, through the activation of internal signaling networks, affect other cells' behavior. This form of cellular communication endows populations of cells with the ability to synchronize their actions. In turn, the ability of single cells to synchronize their decisions may lead to populations of cells to overcome individual cellular limitations to processing environmental information [12, 3, 14]. A natural

question is then how one could quantify these aspects of cellular information-processing [16]

One of the possible approaches, inspired by computer science and automata theory, is membrane computing that propose a formal mathematical framework to investigate the computing power of living cells [15]. We briefly revise here a specific membrane computing model based on agents that can process information internally or in coordination with other cells, highlighting the essential point that higher computing power can only be obtained when the cells are able to coordinate (when individual cells are sufficiently simple). However, an important aspect that is generally not considered in membrane computing (and similar biologically-inspired models of cellular computation) is that a population of cells is subject to Natural selection - cells compete for resources and different cells can replicate at different rates.

This leads to a key problem. Distributed cellular computing relies on the coordination of the different cells in the population by endogenously produced shared molecules. Since these molecules are thus public goods, their production can suffer from the spreading of cheating mutants: cells that benefit from the public good (in this case the enhanced powers of information processing achieved by the population), but which do not contribute to its production [11]. Because these such non-producing cheating cells enjoy the benefits but avoid paying the metabolic costs associated to the production and secretion of the signaling molecules, they have an advantage over the producer cells and will divide faster, spreading in the population and ultimately collapsing the coordination mechanism (and the overall cellular computation). This paper briefly reviews the relevant aspects of this issue and tries to suggests that the bridge between membrane computing and evolutionary dynamics could lead to interesting insights on this problem and, more in general, on the evolutionary stability of cellular computing.

2 A Colony of Synchronizing Agents (CSA) computational framework of cellular synchronization

The issue of evolutionary resilience as described above appears independent of the specific mathematical formalism; however, to provide a simple example of how one could quantify the role of cellular synchronization (from a computational perspective) we briefly review a model of membrane computing called CSA (colony of synchronizing agents) inspired by the interactions between living cells [1] (clearly, the questions driving this paper could also be reformulated in other frameworks of biological computing and cellular decision-making [16, 14]).

The model CSA abstracts intracellular and intercellular mechanisms of cellular populations in terms of multisets-rewriting (often used to analyze the computational aspects of biochemistry [22]). As several models in membrane computing, it is based on a multiset of agents (cells) in a common environment. Each agent has a local contents, stored in the form of a multiset of atomic ob-

jects, updated by multiset rewriting rules which may act on individual agents (intracellular action) or synchronize the contents of pairs of agents (intercellular action). Intercellular actions are abstractions of the process in which cells change synchronously their contents by using a shared environmental molecules (using a terminology from game theory [11], such shared molecule could constitute a public good).

A general approach to quantify the computation power of a system is to compare it with an appropriate device from formal language theory and multiset-rewriting.

In this paper we use only basic aspects of this theory - a more complete introduction can be found in the corresponding book chapter of the membrane computing handbook [15] and in the introductory book on automata theory [8].

The computational model (called CSA [1]) is based on a population/colony of *agents* (e.g., corresponding to *cells*) in a common environment, able to modify their contents and to synchronize with other agents in the same environment. Each agent has a contents represented by a multiset of atomic objects (e.g., corresponding to chemical compounds or the characteristics of individual molecules) with some of the objects classified as terminals (e.g., corresponding to properties or chemicals visible to an external observer). An agent's contents may be modified independently of other agents by means of multiset rewriting rules (called *internal rules*) which can mimic biochemistry or other types of *intracellular mechanisms*. Moreover, the agents can influence each other by synchronously changing their contents using pairwise *synchronization rules* (this represents the ability of the cells to coordinate). Rules are global, so all agents obey the same rules: the only feature which may distinguish the agents is their contents. Dynamics of CSAs are defined as sequences of transitions obtained by applying the rules to the agents (i.e., cells). These transitions thus mark the passage of the system from one configuration to another.

To evaluate its computing power, one can interpret CSAs as computational devices and can thus study CSAs by applying tools from classical fields of computer science, such as formal language and automata theory. This is usually done by defining computations of CSAs the trajectories that reach halting configurations, i.e. configurations where the contents of the agents can no longer be changed because no rules may be applied (this situation can be interpreted as a particular kind of steady state of the system). We are interested in the configuration of the colony when a halting condition is reached and we may take the precise contents of the agents as the output (the result) produced by the CSA.

The model has similarities with cellular automata (CAs) another computational formalism widely used to simulate and model cellular populations [9]. In CAs, cells exist on a regular grid, where each cell has a finite number of possible states and where cells react to or with a defined neighbourhood. In the presented model, because of the multiset-based contents and because of the arbitrary multiset rewriting rules, a cell may have infinitely distinct states and could interact with an arbitrary number of other cells.

From a computational point of view, cellular automata that use synchronous update are computational complete (i.e., equivalent to Turing machines), even when employing *simple* rules (e.g., rule 110 [9]).

In our case, to characterize the computing power of the presented CSAs and describe their dynamics we use few basic concepts from formal language theory and multisets rewriting.

We denote by $|A|$ the cardinality of set A and by $\emptyset$ the empty set. By V^* we denote the set of all strings (sequences of symbols) over V. By V^+ we denote the set of all strings over V excluding the empty string. The empty string is denoted by λ. The *length* of a string v is denoted by $|v|$. The concatenation of two strings $u, v \in V^*$ is written uv.

The number of occurrences of the symbol a in the string w is denoted by $|w|_a$. We denote by $\mathbb{N}$ the set of natural numbers and use standard set operations union, intersection and inclusion denoted by $\cup$, $\cap$ and $\subseteq$, respectively.

The Parikh vector (also called Parikh image) associated to a string x, with respect to an alphabet V, is denoted by $Ps_V(x)$. We then denote by $Ps_V(L)$ the Parikh image of L, with respect to the alphabet V.

Given a multiset M, we denote by $M(a)$ the multiplicity (i.e., number of occurrences) of the symbol a in the multiset M. We denote by $card(M)$ the cardinality of the multiset M.

For multisets M and M' we write $(M \subseteq M')$ to denote that M is included in M'. The *sum* of multisets M and M' is written as the multiset $(M + M')$ and the *difference* between M and M' is written as $(M - M')$. We denote by $\mathbb{M}(V)$ the set of all possible multisets over V and by $\mathbb{M}_m(V)$ the set all multisets over V having cardinality m.

As originally defined in [1], a *Colony of Synchronizing Agents* (CSA) of degree m is a construct $\Pi = (A, T, C, R)$ in which

• A is a finite alphabet of symbols (its elements are called *objects*). $T \subseteq A$ is the set of *terminal objects*.

• An *agent* over A is a multiset over the alphabet A (an agent can be represented by a string $w \in A^*$, since A is finite). C is the *initial configuration of* Π and it is a multiset of agents, with $card(C) = m$.

• R is a finite set of *rules* over A. We have *internal rules* of type $u \to v$, with $u \in A^+$ and $v \in A^*$, and *synchronization rules* of the type $\langle u, v\rangle \to \langle u', v'\rangle$ with $uv \in A^+$ and $u', v' \in A^*$.

An occurrence γ of an internal rule $r : u \to v$ can be applied to an agent w by taking a multiset u from w (hence, $u \subseteq w$) and *assigning* it to γ (i.e., assigning the occurrences of the objects in u to γ). The application of an occurrence of rule r to the agent w consists of removing from w the multiset u and then adding the multiset v to the resulting multiset.

An occurrence γ of a synchronization rule $r : \langle u, v\rangle \to \langle u', v'\rangle$ can be applied to the pair of agents w and w' by: (i) taking from w a multiset u (hence, $u \subseteq w$) and *assigning* it to γ; (ii) taking from w' a multiset v (hence, $v \subseteq w'$) and *assigning* it to γ. The application of an occurrence of rule r to the agents w and w' consists of: (i) removing the multiset u from w and then adding the multiset

u' to the resulting multiset; (ii) removing the multiset v from w' and then adding the multiset v' to the resulting multiset.

Adopting a simplification often used in the area of membrane computing, we assume the existence of a *global clock* which marks the passage of units of time.

A *configuration* of a CSA, Π, consists of the agents present in the colony at a given time. We denote by $\mathbb{C}(\Pi)$ the set of *all possible configurations* of Π. Therefore, using the notation introduced before, $\mathbb{C}(\Pi)$ is exactly $\mathbb{M}(H)$ with $H = \mathbb{M}(A)$.

A single *transition* of Π from an arbitrary configuration c of Π to the next one lasts exactly one time unit and is obtained by applying the rules in the set R to the agents present in c in an *asynchronous* way. This means that, for each agent w and each pair of agents w' and w'' present in c, the occurrences of the objects of w, w' and w'' are *either* assigned to occurrences of the rules, with the occurrences of the objects and the occurrences of the rules chosen in a non-deterministic way, *or* left unassigned. A single occurrence of an object may only be assigned to a single occurrence of a rule. In a transition any number of occurrences of rules (zero, one, or more) can be applied to the agents in the configuration c.

A sequence (possibly infinite) $\langle C_0, C_1, \cdots, C_i, C_{i+1}, \cdots \rangle$ of configurations of Π, where C_{i+1} is obtained from C_i, $i \geq 0$, by a transition is called a *trajectory* of Π. A trajectory of Π is said to be *halting* if it halts, that is if it is finite and the last configuration of the sequence is a *halting configuration*, i.e., a configuration containing only agents for which no occurrences of rules from R can be applied.

A trajectory of Π that is halting and that *starts with the initial configuration* of Π is called a *computation* of Π. The *result/output* of a computation is the *set of vectors of natural numbers*, one vector for each agent w present in the halting configuration with the vector describing the multiplicities of terminal objects present in w. More formally, the result of a computation which stops in the halting configuration C_h is the set of vectors of natural numbers $\{Ps_T(w) \mid w$ is an agent present in $C_h\}$.

Because of the way rules can be applied, several possible computations of Π may exist. Taking the union of all the results for all possible computations of Π, we get the *set of vectors generated* by Π, denoted by $Ps_T(\Pi)$ (that, informally, is what the colony of cells "compute").

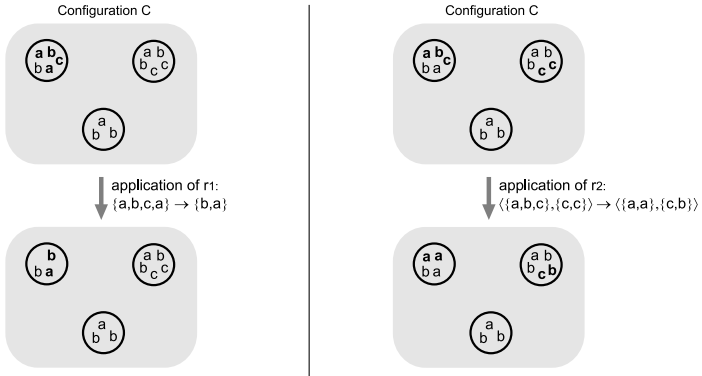
Example 1. A CSA with degree 3 is defined by the following.

$\Pi = (A, T, C, R)$ with $A = \{a, b, c\}$, $T = \{a\}$, $C = \{(abcba, 1), (abbcc, 1), (bab, 1)\} = \{abcba, abbcc, bab\}$.

The rules $R = \{r_1 : abca \to ba,\ r_2 : \langle abc, cc\rangle \to \langle aa, cb\rangle\}$.

The application of an occurrence of internal rule r_1 to the agent *abcba* in the configuration C is shown diagrammatically in Figure 1(a).

The application of an occurrence of the synchronization rule r_2 to the pair of agents *abcba* and *abbcc* in the configuration C is shown diagrammatically in Figure 1(b).



(*a*) Internal rule r_1 applied to C | (*b*) Synchronization rule r_2 applied to C

Fig. 1. Alternative application of rules r_1 and r_2 to configuration C from Example 1.

A more complex example of part of a trajectory is presented in Figure 2(a): $\Pi' = (A', T', C', R')$ with the initial configuration $C' = \{(ac, 2), (a, 1)\}$ and rules $R' = \{ac \to aa,\ a \to b,\ \langle aa, aa\rangle \to \langle ab, ab\rangle,\ \langle ab, d\rangle \to \langle bb, d\rangle,\ b \to d\}$.

In the next Example we show how the output (result) produced by a CSA is obtained.

Example 2. Consider a CSA $\Pi = (A, T, C, R)$ with $A = \{a, b, c, d, e, f\}$, $T = \{e, f\}$, $C = \{(ab, 1), (bc, 1), (bd, 1), (a, 1)\}$. The rules in R are
$\{r_1 : \langle ab, bc\rangle \to \langle eff, eff\rangle, r_2 : \langle ab, bd\rangle \to \langle eff, eff\rangle\}$.

There are *only two possible computations* of Π and these are represented diagrammatically in Figure 2(b).

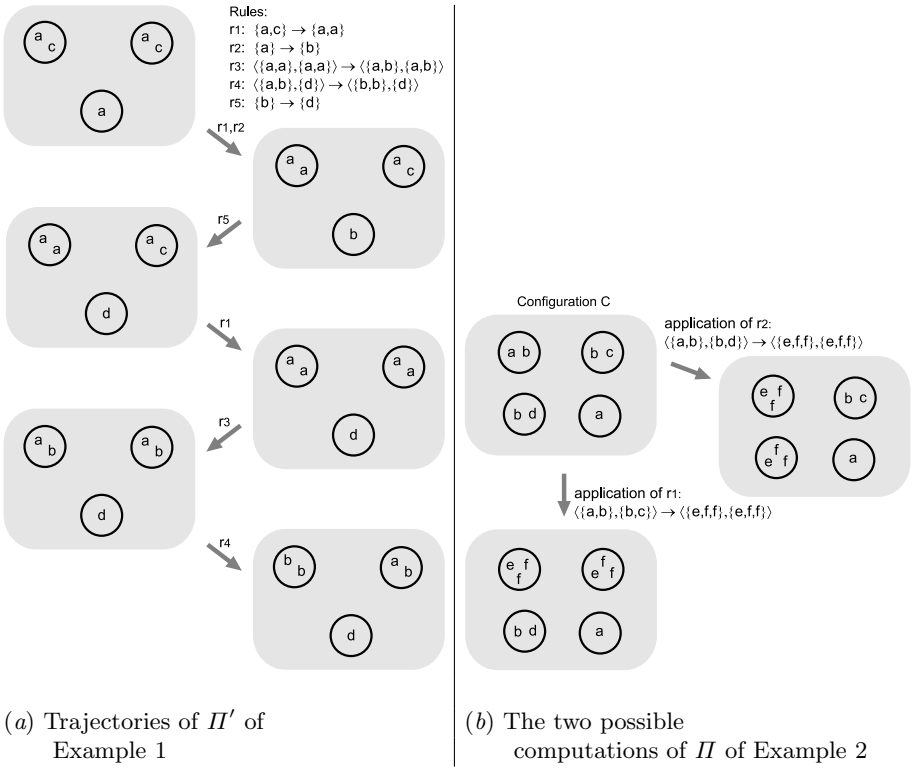
In this case, we have that the output of the system is $Ps_T(\Pi) = \{(1, 2), \overline{0}\}$. Informally, this is what the colony of cells Π computes (a set of vectors of natural numbers).

In fact, we have two possible halting configurations (for the two computations).

This outcome can be understood in the following manner. In the first halting configuration we have the agent (in two copies) eff whose associated Parikh vector (with respect to T) is $(1, 2)$ and the agents bd and a, whose associated Parikh vectors (with respect to T) are null vectors $\overline{0}$ (these agents do not contain any terminal object from T). Then the result of this computation is the set of vectors $\{(1, 2), (1, 2), \overline{0}, \overline{0}\} = \{(1, 2), \overline{0}\}$ with each vector describing the multiplicities of the terminal objects in the agents in the halting configuration.

In the second halting configuration we have the agent (in two copies) eff whose associated Parikh vector (with respect to T) is $(1, 2)$ and the agents bc and a, whose associated Parikh vectors (with respect to T) are null vectors. Then, also in this case, the result of the computation is the set of vectors $\{(1, 2), \overline{0}\}$

Taking the union of the results for the (two) possible computations we get the overall output $Ps_T(\Pi) = \{(1, 2), \overline{0}\} \cup \{(1, 2), \overline{0}\} = \{(1, 2), \overline{0}\}$.



(*a*) Trajectories of Π' of Example 1

(*b*) The two possible computations of Π of Example 2

Fig. 2. Trajectories and computations.

The examples show a specific instance of a CSA but one can study the general computing power of this class of devices. How much CSAs can compute (i.e., assuming one could design/program arbitrary internal and synchronizing rules)?

As proved in [1], if the internal rules of the cells are sufficiently restricted, then the ability for the cells to coordinate (i.e., to employ synchronizing rules) is crucial to get high computational power, i.e., to simulate powerful computing devices such as register machines and Turing machines. In fact, when cells can coordinate, one can formally prove that CSAs are equivalent to a specific class of register machines (called blind [15]), i.e., they can be programmed to compute whatever blind register machines can do. On the other hand, if cells are unable to coordinate then their overall computing power is much more limited [1].

3 The Evolutionary Instability of (Distributed) Cellular Computation

Cellular populations are able to process more complex environmental information by synchronizing or coordinating their responses. As presented in the previous section, this can be formally shown using an abstract agent based model.

Synchronization is a key ingredient of this model, allowing the population to simulate the computations of register machines. Similar results that stress the importance of synchronization, i.e., coordination, between a population of cells/agents can be found for other models of membrane computing [15] and bio-inspired computation [10].

In many biological scenarios, synchronization is essentially based on the possibility to communicate either trough a shared environment or using diffusible signaling molecules [16, 11].

In this last case, at least some of the cells must be able to produce and secrete the diffusible molecule. Such producers are cooperators paying an individual cost to produce a public good: a signal that allows the cells to coordinated their response to a stimulus. The overall computing power of the population is then in the hands of those cells that choose to contributing. However, the public goods nature of the computation is potentially threatened by the endogenous appearance of cheaters or free-riders, cells that take advantage of the enhanced computing power that is mediated by public good, but which do not contribute to its production, avoiding paying the costs associated to it. In situations where the cost-benefit breakdown is favorable to these cheaters, cheating cells may ultimately spread in the population disrupting the ability of the cells to properly process environmental information [11].

In summary, when the ability to perform powerful computations is based on cellular coordination, within-group competition may make cellular coordination inherently evolutionary unstable. One could then hypothesize an endogenous evolutionary cost for distributed cellular computing which would be on top of the energy cost for cellular computations [13]. Given this cost, one may then ask how cellular information-processing, and more generally cellular computing, could be distributed among cells so that it is evolutionary resilient to cheaters.

One of the possibilities is the presence of an interplay between cellular information processing and the eco-evolutionary dynamics that characterize the cellular population, as presented in recent works [2, 4]. Proper cellular information-processing could minimize the risk of cheaters spreading when coupled to specific ecological constraints. This is reminiscent of the connection between optimal decision-making and the particular environment where they are to be applied, as discussed by H.A. Simon [20].

This possibility was investigated computationally by Cavaliere and Poyatos in [2]. In this paper, the authors analyze communities of cells structured in colonies that recurrently are recreated and whose growth depends on a public good (as in the standard Hamilton's group model [21]). The emergence (by mutation) of free-riders (cheaters) that do not contribute but use the accessible public good leads to the demographic collapse of the population. The authors also found that low densities (following the spreading of cheaters and lack of public good) can facilitate the recover of cooperation (i.e., of producer cells) due to a phenomenon called Simpson's paradox [21], but on the other hand they may also lead to demographic extinctions [2]. The risk of extinction can be minimized when the cells use proper cellular information-processing, whereby the amount

investment in the public good can be adjusted in response to the structure of the population, i.e. the fraction of other cooperators in it.

Results in [2] (and presented in Figure 3) show that the optimal kinds of cellular information-processing depend on the ecological constraints (the size of the colonies) and on the efficiency of the public good. For low efficiency of the public good (and sufficiently small colony size) the optimal cellular decision-making consists in withholding production of public goods when cheaters are detected above a certain threshold (this is called positive plasticity). The reaction to cheaters invasions is interrupted as result of the feedback between the cellular decision and colonies assortment stressing how limited information processing can be efficient due to the specific ecological structure in which they are applied [20].

However, such type of decision making does not work when the cellular population is structured in large colonies. In this case, negative plasticity is a better strategy. Such strategy maintains the population in a dynamical equilibrium with the largest possible amount of non-producers that minimizes cheaters advantage but is compatible with population growth. In this case cells do not produce public good unless thy need it - population density reaches critical values.

These results were consistent with later analytical work based on the Ecological Public Goods Game (EPGG), which found that strategies akin to negative plasticity can sustain cooperation in a way that is both ecologically and evolutionary resilient [17]. In this work, Rauch et al. considered an implementation of the Ecological Public Goods Game [6, 7], where cooperators were allowed to implement either fixed strategies with different investment levels, or facultative strategies where the amount of public good that cooperators produced could be tuned in response to the density of cooperators in the population. These authors found that the structure of the EPGG causes an inherent tradeoff between ecological resilience (the ability to recover from perturbations that cause a sudden increase in mortality) and evolutionary resilience (the ability to resist invasion by cheaters). Cooperators that use a (typically low) fixed investment strategy cannot avoid this tradeoff, and therefore strategies that are non-invasible by cheaters have low ecological resilience and are prone to extinction in a fluctuating environment. On the other hand, cooperators that are highly stable to such ecological stressors (by producing large amounts of the public good) were very sensitive to invasion by cheaters, which would take over the population and drive cooperators to exceedingly low frequencies.

However, cooperators that used facultative rather than fixed investments were able to circumvent this tradeoff and reach high levels of ecological resilience while remaining fully resistant to invasion by cheaters. The facultative strategies that yielded this behavior were reminiscent of the negative plasticity strategies that Cavaliere and Poyatos found stabilized cooperation in agent based simulations. The two studies differ in the exact nature of the facultative or plastic strategies. In the EPGG-based analytical model, the amount of investment (rather than the probability to invest or not, as was the case for the computational model) was modulated in the facultative strategy. In addition, the

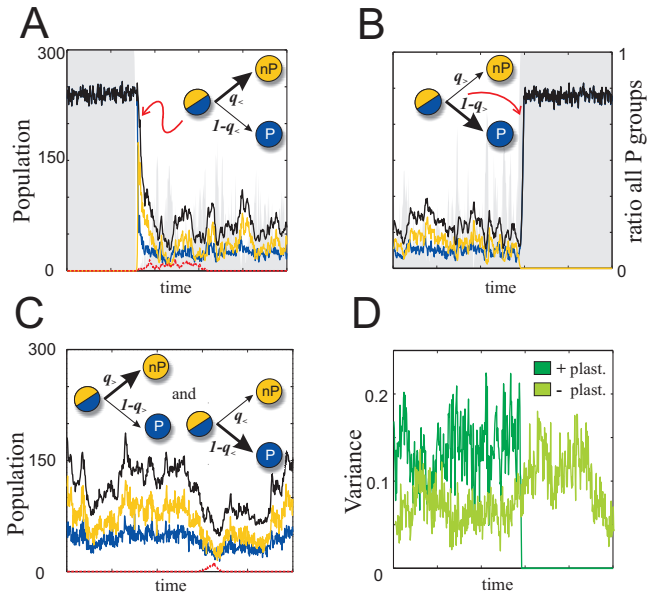


Fig. 3. Cellular decision-making can repress cheating cells. Cellular information-processing can alter the evolutionary dynamics and constrain the spreading of cheaters in communities of cells structured in colonies, i.e., Hamilton's group model [21] (Figures from [2]). Tick arrows denote the cellular decision to switch to production (P) or non production (nP) of the public good; q denotes the individual chances to switch to nP given the amount of public good received. (A) Positive plasticity consists in producing public good, stopping it as response to cheaters. This causes a fast reduction of the amount of public good and consequently of population density. After cheater invasion is stopped, the population finally evolve to the initial scenario of all cells producing (B). (C) A population of negative plastic producers is characterized by its permanent low density favored by the constant presence of non-producing cells which helps controlling cheaters invasion. (D) The success of the decision-making is coupled to the ecological constraints as it is related to the colonies heterogeneity, i.e., positive plasticity transiently modifies the inter-colonies variance to control cheaters. This contrasts with the relatively constant variance observed in a population of negative plastic producers (variances correspond to time series (B) and (C), respectively).

analytical model consider infinitely large populations and produced deterministic dynamics, whereas the computational models considered finite populations, and thus stochastic dynamics. Agent based simulations with large population sizes displayed behavior that was consistent with the analytical model, thus bridging the two approaches.

Altogether, these two studies showed that if cellular information-processing is properly coupled to environmental constraints, then cheaters spreading can be minimized, public goods production can be maintained, and this may facili-

tate the presence of cellular coordination (hence, more complex kinds of cellular computations, as discussed above).

Beyond the realm of theoretical speculation, the basic types of cellular decision-making outlined above (negative or positive plasticity) are often found in natural scenarios [4]. Intriguingly, the types of strategies used by microbes in nature are consistent with the strategies predicted by mathematical and computational models to stabilize cooperation. Negative plasticity resembles the notion of facultative cheating, a cellular strategy implemented by different molecular mechanisms: generation of iron-scavenging pyoverdin molecules –iron being an essential public good in some environments– is reduced when enough molecules are already in the environment minimizing in this way the ability of cheaters to invade [5]. On the other hand, gene regulatory functions that are consistent with positive plasticity have also been documented in cases where eco-evolutionary modeling would predict they would maximize resistance to cheater invasion [19, 18]

4 Perspective

Drawing inspirations from biology, computer science, mathematics and engineering one can understand, study and quantify the process of cellular decision-making as the ability of cells to process environmental information and take decisions (i.e., compute) in a distributed manner [10, 16]. There are multiple ways to quantify how much cells can compute. Membrane computing is a computational model inspired by the principles of information-processing in living cells and focused on automata theory [15]. Several results in the area of membrane computing highlight, in various forms, a key point: The ability to encode powerful computations is generally linked to the ability of the cells to have some sort of synchronization or coordination of their responses i.e., the ability to communicate and to exchange (simple) messages [15]. This is, for instance, the case in the simple model of agents (abstracted cells) [1] that we have briefly reviewed in this paper - the ability of agents to synchronize allow the population to perform powerful computations, equivalent to a class of register machines.

One usual way to facilitate synchronization between cells is the sharing of a diffusible signaling molecule produced by cooperative (producer) cells [16]. These molecules may act as a public good, and lead to a classical evolutionary problem: cheating cells that avoid the costly production of the public good can appear by mutation and spread in the population, causing a concomitant decline in the public good [11]. In this sense, then, we could say that a corollary of this is that population of living cells find challenges to parallelize complex computations, as the need for synchronization may lead to the appearance of cheaters that would in turn lead to the collapse of the computation. From this point of view, one could even speculate that the complexity of distributed cellular computation is bounded by the ability to encode complex functions and the need to be evolutionary sustainable.

On the other hand, the ability of individual cells to process information can itself be a possible solution to the cheating problem. This is indeed the case when such information-processing is coupled to the ecological constraints in which the cellular population is embedded. This possibility discussed in [2] and [4], however, has not been studied in a systematic way and the area of membrane computing may be a possible framework in which one can easily combine the notion of cellular computation with evolutionary dynamics.

For instance, one could quantitatively study the potential trade-off between cellular computation, parallelism and evolutionary resilience. How can a computational process be distributed between individual cells and which is the cost to pay in terms of evolutionary resilience ? Which is the best strategy to divide the computational labor between the cells to avoid the emergence of detrimental mutants ?

All these types of questions may be approached by extending standard models of membrane computing. For instance, in the context of the presented model with agents, one may incorporate cell division and assume that cells may divide at different speed - cells that are simpler (in terms of the employed internal biochemistry/computational instructions) will divide faster (i.e., possibly spreading in the population, altering the balance between different types of cells). This would couple the notion of cellular fitness to that of cellular computation. Such idea could also be incorporated in classical models of membrane computing with division, known for their ability to solve efficiently (in parallel way) hard computational problems [15]. One could then study which of the obtained results with membrane systems with divisions are also evolutionary stable - i.e., they are not compromised by the fact that some cells may execute less complex instructions/biochemistry and then divide faster (remains to formulate a proper measure for the complexity of instructions).

We believe that these types of questions can help to address one of the key differences between distributed cellular computing and standard models of distributed computing - cells need to split the computational tasks with the evolutionary constraint to keep the proper balance between the different types, avoiding that some of them would outcompete the others.

Acknowledgments M.C. acknowledges the support from the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J02175X/1. Work in the Sanchez laboratory is supported by a Young Investigator grant from the Human Frontiers Science Project and a Scialog seed grant from Simons Foundation.

References

1. M. Cavaliere, R. Mardare, and S. Sedwards. A multiset-based model of synchronizing agents: Computability and robustness. *Theoretical Computer Science*, 391(3):216–238, feb 2008.
2. M. Cavaliere and J. F. Poyatos. Plasticity facilitates sustainable growth in the commons. *Journal of The Royal Society Interface*, 10(81):20121006–20121006, jan 2013.

3. O. Feinerman and A. Korman. Theoretical distributed computing meets biology: A review. In *International Conference on Distributed Computing and Internet Technology*, pages 1–18. Springer Berlin Heidelberg, 2013.
4. K. I. Harrington and A. Sanchez. Eco-evolutionary dynamics of complex social strategies in microbial communities. *Communicative & Integrative Biology*, 7(1):e28230, jan 2014.
5. F. Harrison, J. Paul, R. C. Massey, and A. Buckling. Interspecific competition and siderophore-mediated cooperation in pseudomonas aeruginosa. *The ISME journal*, 2(1):49–55, 2008.
6. C. Hauert, M. Holmes, and M. Doebeli. Evolutionary games and population dynamics: maintenance of cooperation in public goods games. *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1600):2565–2571, 2006.
7. C. Hauert, J. Y. Wakano, and M. Doebeli. Ecological public goods games: cooperation and bifurcation. *Theoretical population biology*, 73(2):257–263, 2008.
8. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
9. A. Ilachinski. *Cellular automata: a discrete universe*. World Scientific, 2001.
10. L. Kari and G. Rozenberg. The many facets of natural computing. *Communications of the ACM*, 51(10):72, oct 2008.
11. S. A. Levin. Public goods in relation to competition, cooperation, and spite. *Proceedings of the National Academy of Sciences*, 111(Supplement_3):10838–10845, jul 2014.
12. J. Macía, F. Posas, and R. V. Solé. Distributed computation: the new wave of synthetic biology devices. *Trends in Biotechnology*, 30(6):342–349, jun 2012.
13. P. Mehta and D. J. Schwab. Energetic costs of cellular computation. *Proceedings of the National Academy of Sciences*, 109(44):17978–17982, 2012.
14. S. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015.
15. G. Paun, G. Rozenberg, and A. Salomaa. *The Oxford handbook of membrane computing*. Oxford University Press, Inc., 2010.
16. T. J. Perkins and P. S. Swain. Strategies for cellular decision-making. *Mol Syst Biol*, 5, nov 2009.
17. J. Rauch, J. Kondev, and A. Sanchez. A tradeoff between the ecological and evolutionary stabilities of public goods genes in microbial populations. *bioRxiv*, 2016.
18. A. Ross-Gillespie, A. Gardner, A. Buckling, S. A. West, and A. S. Griffin. Density dependence and cooperation: theory and a test with bacteria. *Evolution*, 63(9):2315–2325, 2009.
19. A. Ross-Gillespie, A. Gardner, S. A. West, and A. S. Griffin. Frequency dependence and cooperation: theory and a test with bacteria. *The American Naturalist*, 170(3):331–342, 2007.
20. H. A. Simon. Models of man; social and rational. 1957.
21. E. Sober. *The nature of selection: evolutionary theory in philosophical focus*. University of Chicago Press, 1993.
22. D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, feb 2008.